Neuroevolution of augmenting topologies - backgammon

Igor Kandić

September 28, 2024

1 Problem description

1.1 History

Backgammon is a two-player board game played with counters and dice on tables boards. It is the most widespread Western member of the large family of tables games, whose ancestors date back nearly 5,000 years to the regions of Mesopotamia and Persia. The earliest record of backgammon itself dates to 17th-century England, being descended from the 16th-century game of Irish.

1.2 Setup

Each side of the board has a track of 12 long triangles, called points. The points form a continuous track in the shape of a horseshoe, and are numbered from 1 to 24. In the most commonly used setup, each player begins with fifteen pieces; two are placed on their 24-point, three on their 8-point, and five each on their 13-point and their 6-point. The two players move their pieces in opposing directions, from the 24-point towards the 1-point.

Points 1 through 6 are called the home board or inner board, and points 7 through 12 are called the outer board. The 7-point is referred to as the bar point, and the 13-point as the midpoint. Usually the 5-point for each player is called the "golden point".

To start the game, each player rolls one die, and the player with the higher number moves first using the numbers shown on both dice.

1.3 Movement

If the players roll the same number, they must roll again, leaving the first pair of dice on the board. The player with the higher number on the second roll moves using only the numbers shown on the dice used for the second roll. Both dice must land completely flat on the right-hand side of the gameboard. The players then take alternate turns, rolling two dice at the beginning of each turn.

After rolling the dice, players must, if possible, move their pieces according to the number shown on each die. For example, if the player rolls a 6 and a 3 (denoted as "6-3"), the player must move one checker six points forward, and another or the same checker three points forward. The same checker may be moved twice, as long as the two moves can be made separately and legally: six and then three, or three and then six. If a player rolls two of the same number, called doubles, that player must play each die twice. For example, a roll of 5-5 allows the player to make four moves of five spaces each. On any roll, a player must move according to the numbers on both dice if it is at all possible to do so. If one or both numbers do not allow a legal move, the player forfeits that portion of the roll and the turn ends. If moves can be made according to either one die or the other, but not both, the higher number must be used. If one die is unable to be moved, but such a move is made possible by the moving of the other die, that move is compulsory.

In the course of a move, a checker may land on any point that is unoccupied or is occupied by one or more of the player's own checkers. It may also land on a point occupied by exactly one opposing checker, or "blot". In this case, the blot has been "hit" and is placed in the middle of the board on the bar that divides the two sides of the playing surface. A checker may never land on a point occupied by two or more opposing checkers; thus, no point is ever occupied by checkers from both players simultaneously. There is no limit to the number of checkers that can occupy a point or the bar at any given time. Checkers placed on the bar must re-enter the game through the opponent's home board before any other move can be made. A roll of 1 allows the checker to enter on the 24-point (opponent's 1), a roll of 2 on the 23-point (opponent's 2), and so forth, up to a roll of 6 allowing entry on the 19-point (opponent's 6). Checkers may not enter on a point occupied by two or more opposing checkers. Checkers can enter on unoccupied points, or on points occupied by a single opposing checker; in the latter case, the single checker is hit and placed on the bar. A player may not move any other checkers until all checkers belonging to that player on the bar have re-entered the board.

If a player has checkers on the bar, but rolls a combination that does not allow any of those checkers to re-enter, the player does not move. If the opponent's home board is completely "closed" (i.e. all six points are each occupied by two or more checkers), there is no roll that will allow a player to enter a checker from the bar, and that player stops rolling and playing until at least one point becomes open (occupied by one or zero checkers) due to the opponent's moves.

A turn ends only when the player has removed his or her dice from the board. Prior to this moment, a move can be undone and replayed an unlimited number of times.

1.4 Bearing off

When all of a player's checkers are in that player's home board, that player may start removing them; this is called "bearing off". A roll of 1 may be used to bear off a checker from the 1-point, a 2 from the 2-point, and so on. If all of a player's checkers are on points lower than the number showing on a particular die, the player must use that die to bear off one checker from the highest occupied point. For example, if a player rolls a 6 and a 5, but has no checkers on the 6-point and two on the 5-point, then the 6 and the 5 must be used to bear off the two checkers from the 5-point. When bearing off, a player may also move a lower die roll before the higher even if that means the full value of the higher die is not fully utilized. For example, if a player has exactly one checker remaining on the 6-point, and rolls a 6 and a 1, the player may move the 6-point

checker one place to the 5-point with the lower die roll of 1, and then bear that checker off the 5-point using the die roll of 6; this is sometimes useful tactically. As before, if there is a way to use all moves showing on the dice by moving checkers within the home board or by bearing them off, the player must do so. If a player's checker is hit while in the process of bearing off, that player may not bear off any others until it has been re-entered into the game and moved into the player's home board, according to the normal movement rules.

The first player to bear off all fifteen of their own checkers wins the game. When keeping score in backgammon, the points awarded depend on the scale of the victory. A player who bears off all fifteen pieces when the opponent has borne off at least one, wins a single game worth 1 point. If all fifteen have been borne off before the opponent gets at least one checker off, this is a gammon or double game worth 2 points. A backgammon or triple game is worth 3 points and occurs when the losing player has borne off no pieces and has one or more on the bar and/or in the winner's home table (inner board).

1.5 Doubling cube

To speed up match play and to provide an added dimension for strategy, a doubling cube is usually used. The doubling cube is not a die to be rolled, but rather a marker, with the numbers 2, 4, 8, 16, 32, and 64 inscribed on its sides to denote the current stake. At the start of each game, the doubling cube is placed on the midpoint of the bar with the number 64 showing; the cube is then said to be "centered, on 1". When the cube is still centered, either player may start their turn by proposing that the game be played for twice the current stakes. Their opponent must either accept ("take") the doubled stakes or resign ("drop") the game immediately.

Whenever a player accepts doubled stakes, the cube is placed on their side of the board with the corresponding power of two facing upward, to indicate that the right to redouble, which is to offer to continue doubling the stakes, belongs exclusively to that player. If the opponent drops the doubled stakes, they lose the game at the current value of the doubling cube. For instance, if the cube showed the number 2 and a player wanted to redouble the stakes to put it at 4, the opponent choosing to drop the redouble would lose two, or twice the original stake.

There is no limit on the number of redoubles. Although 64 is the highest number depicted on the doubling cube, the stakes may rise to 128, 256, and so on. In money games, a player is often permitted to "beaver" when offered the cube, doubling the value of the game again, while retaining possession of the cube.

This paper describes a unsuccessful approach toward a computer program that plays a good game of backgammon. Because backgammon incorporates a stochastic process, the roll of the dice, the game tree space is too large to lend itself to the traditional method of search. For this reason, past attempts at computer programs that played backgammon have been largely unsuccessful. A computer program that succeeded in doing so is a major milestone. The goal of this project was to recreate this major milestone.

2 Method

The trained program should act as both an evaluator and a controller, by utilizing board evaluations to choose that move that will lead to the best value. It will also, therefore, be stateless - the network will need as input only a single board description, and will give the same output for that description regardless of what it has seen before (excepting differences caused by training of the network.) The neural network must therefore be embedded within a game-playing framework to achieve our goal of competent play, because the network by itself cannot choose moves. We have, then, the constraints on program design: A neural network evaluation function, embedded within a framework that ensures correct move choice and provides an interface from which the network can play against others.

The constraints above lead naturally into an overall program architecture. A framework is set up that tracks changes to the board, rolls dice, and ensures the legality of moves. The framework is also responsible for generating the stream of moves that the neural network must evaluate. The framework exhibits no intelligence by itself- it simply finds, and evaluates via the neural network, all legal moves given an initial board position and roll of the dice.

2.1 Neuroevolution of augmenting topologies (NEAT)

NeuroEvolution of Augmenting Topologies (NEAT) is a genetic algorithm (GA) for the generation of evolving artificial neural networks (a neuroevolution technique) developed by Kenneth Stanley and Risto Miikkulainen in 2002 while at The University of Texas at Austin. It alters both the weighting parameters and structures of networks, attempting to find a balance between the fitness of evolved solutions and their diversity. It is based on applying three key techniques: tracking genes with history markers to allow crossover among topologies, applying speciation (the evolution of species) to preserve innovations, and developing topologies incrementally from simple initial structures ("complexifying").

We closely follow implementation from Stanley's for core NEAT features and Sethbling's "MarI/O - Machine Learning for Video Games" for general code architecture of GA. Whole project is implemented in C++.

2.2 Algorithm

Traditionally, a neural network topology is chosen by a human experimenter, and effective connection weight values are learned through a training procedure. This yields a situation whereby a trial and error process may be necessary in order to determine an appropriate topology. NEAT is an example of a topology and weight evolving artificial neural network (TWEANN) which attempts to simultaneously learn weight values and an appropriate topology for a neural network.

In order to encode the network into a phenotype for the GA, NEAT uses a direct encoding scheme which means every connection and neuron is explicitly represented. This is in contrast to indirect encoding schemes which define rules that allow the network to be constructed without explicitly representing every connection and neuron, allowing for more compact representation.

The NEAT approach begins with a perceptron-like feed-forward network of only input neurons and output neurons. As evolution progresses through discrete steps, the complexity of the network's topology may grow, either by inserting a new neuron into a connection path, or by creating a new connection between (formerly unconnected) neurons.

2.2.1 Encoding

NEAT (NeuroEvolution of Augmenting Topologies) employs a sophisticated genetic encoding scheme designed to facilitate the evolution of neural networks through a structured representation of their connectivity. Each genome within NEAT is represented as a linear sequence that delineates the connectivity between nodes, allowing for straightforward manipulation during evolutionary processes such as crossover and mutation.Figure 1

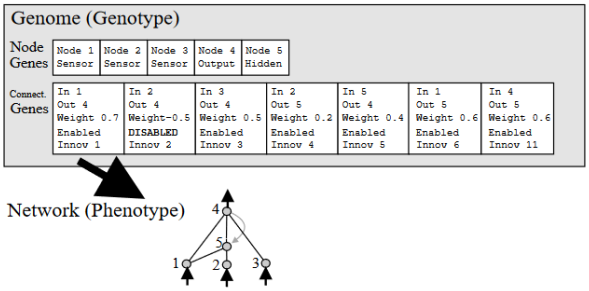

Figure 1: Visual representation of genome

2.2.2 Innovation

Innovation is a cheat that allows us to align matching genes in different genomes. It represents a historical origin of each gene. Two genes with the same historical origin must represent the same structure.

2.2.3 Crossover

Crossover relies on specific encoding and innovation explained earlier. Matching genes are inherited randomly, whereas disjoint genes (those that do not match in the middle) and excess genes (those that do not match in the end) are inherited from the more fit parent. In this case, equal fitnesses are assumed, so the disjoint and excess genes are also inherited randomly.

2.2.4 Mutation

We supported two types of mutation. Addition of new connect gene (edge in phenotype) and addition of new node gene (node in phenotype).Figure 3

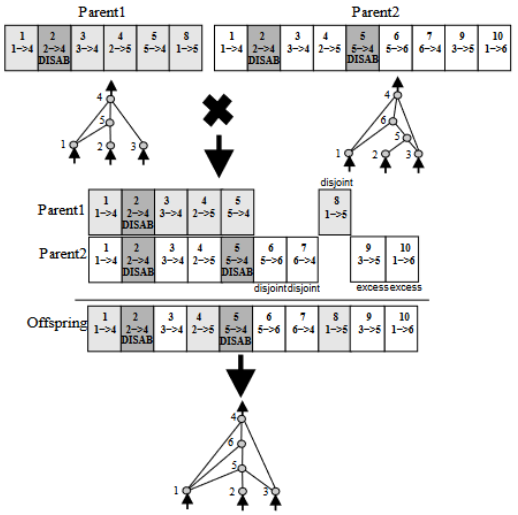

Figure 2: Crossover

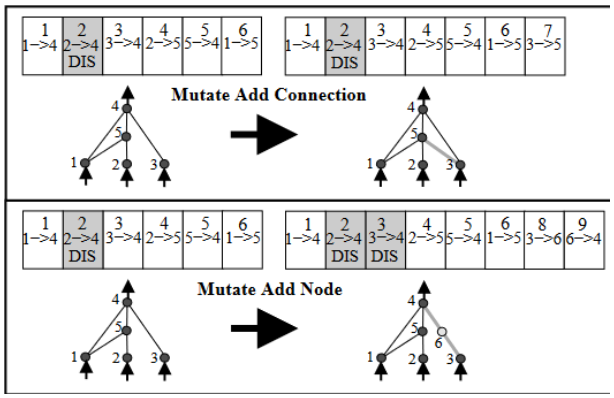

Figure 3: Mutation

2.2.5 Fitness evaluation

Tournament selection is used for breeding selection. Each genome plays n games of backgammon in a round-robin fashion against every other genome in generation. For each win they get 1 point and for loss 0. Total fitness of genome is it's total score.

2.3 Results

Network was trained for 603 generations.
Size of generation was 100 and number of games per opponent was 50.

Plan was 10000 generations with size of 1000 but due to complexity of evaluation neural network and finding all legal moves in a game it was not feasible.

In the end trained network managed to have 70% WR against the initial network that only selected first legal generated move.

3 Conclusion

We think its possible for NEAT to learn to play backgammon but only with the help of GPU. This project was written in C++ so its possible to implement compute intensive things to execute on GPU with CUDA.

4 References

Evolving Neural Networks Through Augmenting Topologies - Kenneth O. Stanley and Risto Miikkulainen
MarI/O - Machine Learning for Video Games - sethbling
Hoyle's rules of games : descriptions of indoor games of skill and chance, with advice on skillful play : based on the foundations laid down by Edmond Hoyle NVIDIA® CUDA® Toolkit